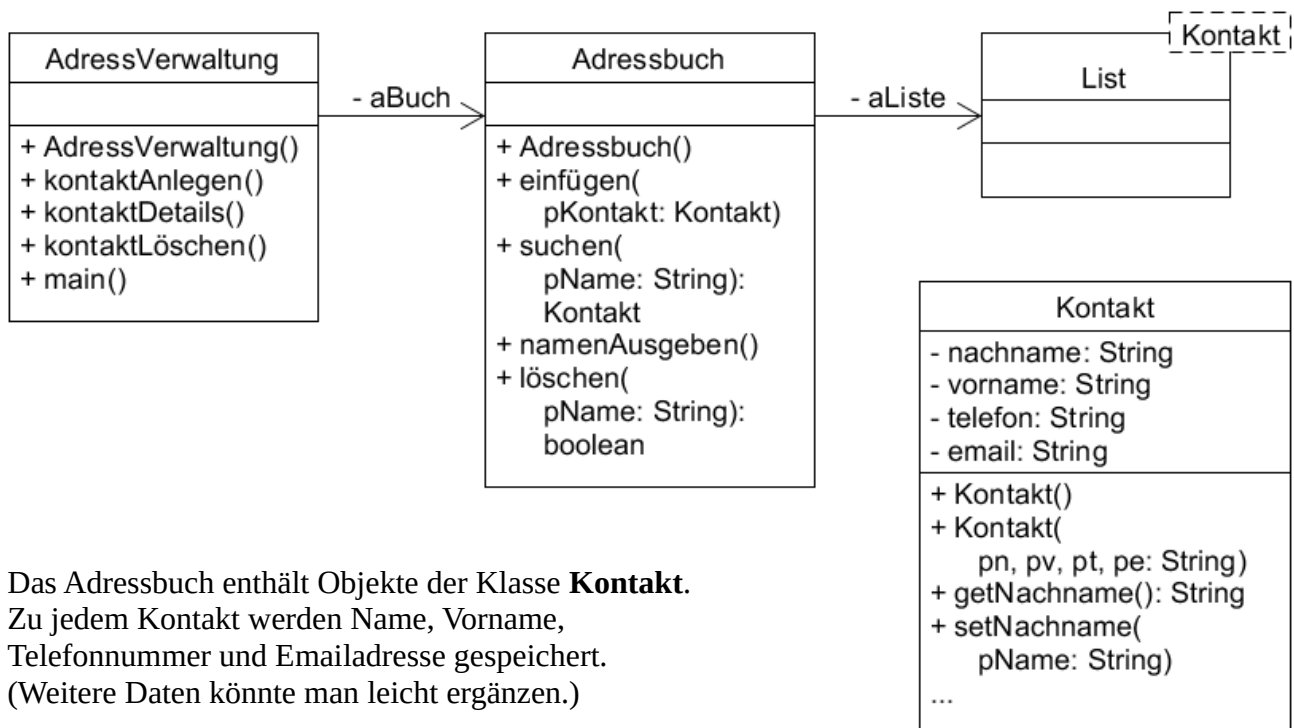


Eine typische Anwendung für eine Liste ist ein **Adressbuch**:
Ein Adressbuch enthält eine Liste von Kontakten.
Jeder Kontakt gehört zu einer Person, und speichert Namen, Adresse, Telefonnummer, Emailadresse usw.

Ein Adressbuch ist in der Regel nicht statisch, d.h. im Lauf der Zeit kommen neue Adressen hinzu, nicht mehr benötigte Kontakte werden gelöscht, und manchmal ändern sich Daten, z.B. wenn jemand umzieht.



Die Software für das Adressbuch ist im folgenden Klassendiagramm modelliert:



Das Adressbuch enthält Objekte der Klasse **Kontakt**.
Zu jedem Kontakt werden Name, Vorname, Telefonnummer und Emailadresse gespeichert.
(Weitere Daten könnte man leicht ergänzen.)

Die Kontakte werden in einer **Liste** gespeichert.

Die Klasse **Adressbuch** bietet Methoden, um Kontakte in die Liste einzufügen und zu suchen. In dieser Klasse werden die Algorithmen mit der Datenstruktur Liste implementiert.

Das Adressbuch soll mithilfe der Konsole bedient werden. Diese Benutzerschnittstelle wird in der Klasse **AdressVerwaltung** (Hauptklasse) implementiert, mit einer **Ausnahme**:

Die Methode namenAusgeben() gibt die Namen aller Kontakte auf der Konsole aus. Die Klasse Adressbuch erlaubt keinen direkten Zugriff auf die Liste. Da die Ausgabe aber Zugriff auf alle Elemente der Liste braucht, lässt sich diese Methode besser in der Klasse Adressbuch implementieren.

Aufgabe 1 (typische Abituraufgabe)

Erläutere für jede der vier Datenstrukturen Array, Queue, Stack und Liste, wie gut diese für die Anwendung „Adressbuch“ geeignet ist bzw. nicht geeignet ist. Gib dabei jeweils Vor- und Nachteile an.

Aufgabe 2

Verwende die ausgeteilte BlueJ-Vorlage. Die Klassen Kontakt und List sind bereits fertig implementiert. Implementiere die Klasse **Adressbuch** nach den folgenden Vorgaben:

- a) `Adressbuch()` (Konstruktor)
Erzeugt die (leere) Liste.
- b) `void einfügen(Kontakt pKontakt)`
Als Parameter `pKontakt` wird ein bereits mit Daten gefülltes Kontakt-Objekt übergeben. Die Methode fügt diesen Kontakt einfach am Ende der Liste ein.
(Das Objekt zu erzeugen und die Daten vom Benutzer zu erfragen ist Aufgabe der Hauptklasse)
- c) `Kontakt suchen(String pName)`
Als Parameter wird ein Nachname übergeben. Die Methode sucht in der Liste das Kontakt-Objekt mit diesem Nachnamen und gibt es zurück. Falls kein entsprechendes Kontakt-Objekt in der Liste vorhanden ist, wird null zurückgegeben.
Wir nehmen zur Vereinfachung an, dass jeder Name nur einmal in der Liste vorkommt.

Zur Erinnerung: Strings werden mit `equals()` verglichen:

```
if (str1.equals(str2)) { ... }
```
- d) `void namenAusgeben()`
Gibt die Nachnamen und Vornamen für alle Kontakte der Liste auf der Konsole aus. Außerdem wird die Anzahl der Kontakte ausgegeben.
- e) `boolean löschen(String pName)`
Die Methode sucht in der Liste das Kontakt-Objekt mit Nachnamen `pName` und löscht dieses, falls es gefunden wird. In diesem Fall gibt die Methode `true` zurück. Falls kein entsprechendes Kontakt-Objekt in der Liste vorhanden ist, wird `false` zurückgegeben.

Aufgabe 3

Implementiere die Hauptklasse **AdressVerwaltung**.

Bei Schwierigkeiten mit der Konsole findest du eine Hilfestellung in der Vorlage.

- a) `AdressVerwaltung()` (Konstruktor)
Erzeugt das Adressbuch-Objekt.
- b) `void kontaktAnlegen()`
Erfragt die Daten für ein neues Kontakt-Objekt (Name, Vorname, Telefon, Email) vom Benutzer. Erstellt das Kontakt-Objekt und fügt es ins Adressbuch ein.
- c) `void kontaktDetails()`
Bittet um die Eingabe eines Nachnamens.
Sucht den entsprechenden Kontakt aus dem Adressbuch und gibt dessen Daten aus.
Falls der gesuchte Name nicht enthalten ist, wird eine entsprechende Meldung ausgegeben.
- d) `void kontaktLöschen()`
Bittet um die Eingabe eines Nachnamens.
Löscht den entsprechenden Kontakt aus dem Adressbuch.
Falls der gesuchte Name nicht enthalten ist, wird eine entsprechende Meldung ausgegeben.
- e) `main()`
Zeigt ein Konsolenmenü an. Der Benutzer wählt einen Menüpunkt durch Eingabe einer Zahl.
Die gewählte Funktion wird ausgeführt.
Das wiederholt sich solange, bis der Benutzer das Programm beendet.

Das Menü könnte folgendermaßen aussehen:

Wählen Sie einen der folgenden Punkte

1. Neuer Kontakt
2. Kontakte auflisten
3. Kontaktdetails
4. Kontakt löschen
5. Programm beenden